Transcending the AAA data access patterns

Matevž Tadel

Talk outline:

1. Introduction: what is this really about
2. Pull out the guts of AAA monitoring data
3. Conclusion?

mtadel@ucsd.edu

The gist

- Detailed monitoring data (ROOT trees):
  - 25 GB of detailed monitoring data from June ’12
    - file-access-reports, more or less the same thing that goes to OSG Gratia and to Dashboard
  - 41 GB of super-detailed data from Feb ’13
    - full detail on all reads (including vector), IOV monitoring option
    - all UCSD servers + one server at UNL
- Nobody every really tried to dive into this
- This should also help to steer:
  - the next generation of computing models
  - implementation and operation of caching-proxies

What's there to transcend??!

- A lot of AAA effort went into monitoring
  - Both to implement it and to use it in CMS
  - Should we (CMS) continue at this level?
  - Should we do something different?
- It's also a personal thing, longing for closure
  - Make a comprehensive review of what can be extracted from the monitoring data.
  - Understand it … ??? … profit!

So, how does one do it?

- There's about 2 months of work in this:
  - wrote 4 or 5 loopers / data extractors
  - at the end made a mini analysis framework
    - which got messy over the last two weeks …
  - it really wasn't trivial
    - and we often give this work to new students, sigh
    - I'm guessing this was actually meant for me.
- The story is as interesting as the results …

1. Data from 10,000 ft
2. Fun facts about it
3. How much of it is really worth looking at? (Hint: littlesome)
4. Results! And more of them …

20 MONTHS OF AAA FILE ACCESS REPORTS

Count users, server/client sites, top directories, data tiers, … files

Fill loads of std::map<TString, struct Accumulate> then sort this by:

- # of accesses
- total transferred data
- total open duration

Plot overview histograms for some sub-selections

STEP I. – WHAT ALL IS IN THIS DATA

AAA **FAR** data overview I.

2012 Jun - 2014 Feb

5.3×10^7s = 14,761 h = 88 weeks = 20.2 months

	N (M)	Vol (PB)
All	199.688	81.162
US	169.924	74.457
local	146.818	62.809
remote	23.105	11.648
US srvs to X	2.252	1.084
X srvs to US	0.273	0.158

matevz@desire xrd-far> ls --format single-column –sh

total 25G
110M xmfar-2012-06.root
528M xmfar-2012-07.root
854M xmfar-2012-08.root
473M xmfar-2012-09.root
940M xmfar-2012-10.root
808M xmfar-2012-11.root
627M xmfar-2012-12.root
846M xmfar-2013-01.root
1.1G xmfar-2013-02.root
1.5G xmfar-2013-03.root
933M xmfar-2013-04.root
747M xmfar-2013-05.root
1.2G xmfar-2013-06.root
1.3G xmfar-2013-07.root
1.3G xmfar-2013-08.root
1.7G xmfar-2013-09.root
1.8G xmfar-2013-10.root
1.9G xmfar-2013-11.root
1.1G xmfar-2013-12.root
1.7G xmfar-2014-01.root
3.5G xmfar-2014-02.root

130 Bytes / record

AAA FAR data overview II.

1. All (200M):

- 91 M unauthenticated; Brian B. 10 M, Andrea S. 5.5 M
- about 1200 different DNs

2a. USA, local access (147M):

- 72 M unauthenticated
- transfers:
  - domains: 60% UW, 23% Purdue, 10% FNAL, 3.5% UNL, 2% MIT
  - tiers: 40% AOD, 23% AODSIM, 10% GEN-SIM, 3.7% GEN-SIM-RECO, 2.7% RECO, 1% RAW

AAA FAR data overview III.

2b. USA remote (23M):

- open counts: about 40% is monitoring / testing / development
- transfers: 6.6% production, 30% 15 top users
  - server: FNAL (60%), UNL (15%), UW, UCSD (10% each)
  - client: UW (25%), Caltech, Purdue, UCSD, UNL, ND (10 - 15%)
  - AOD, AODSIM most read by far (35% each)

    above 5%: GEN-SIM-RAW, RAW, RECO
- Curiosities:
  - the most accessed files (monitoring): 2.6 and 2.1 M-times!; next 830 k

    about 100 files that are accessed more than 1000-times.
  - /store/user/ra2tau 4.7% transfers (450 madgraph pat-skim files, accessed 70-150 times!), two more users around 0.4%

AAA FAR data overview IV.

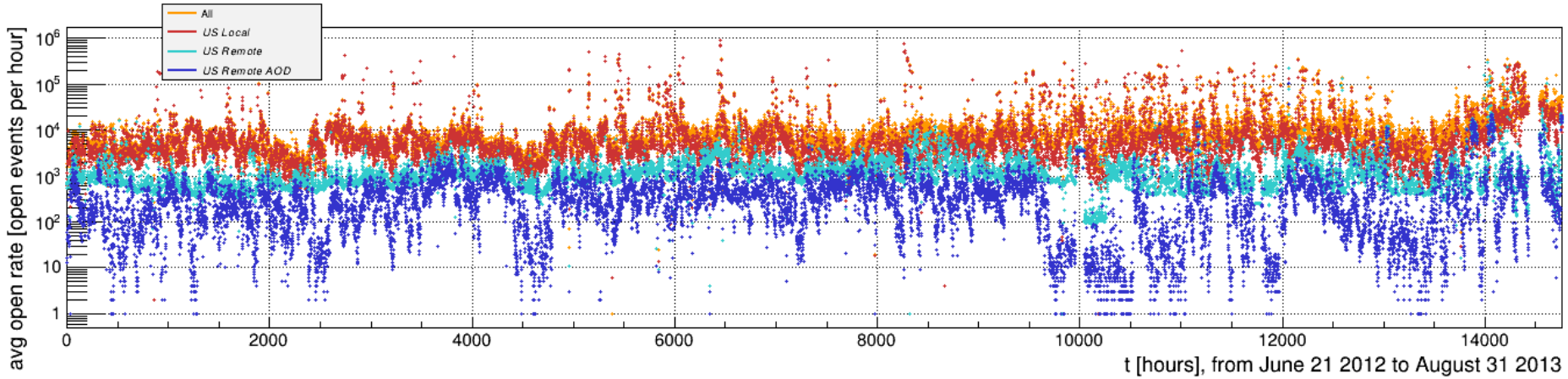
4. Access into USA from elsewhere (2.3M):

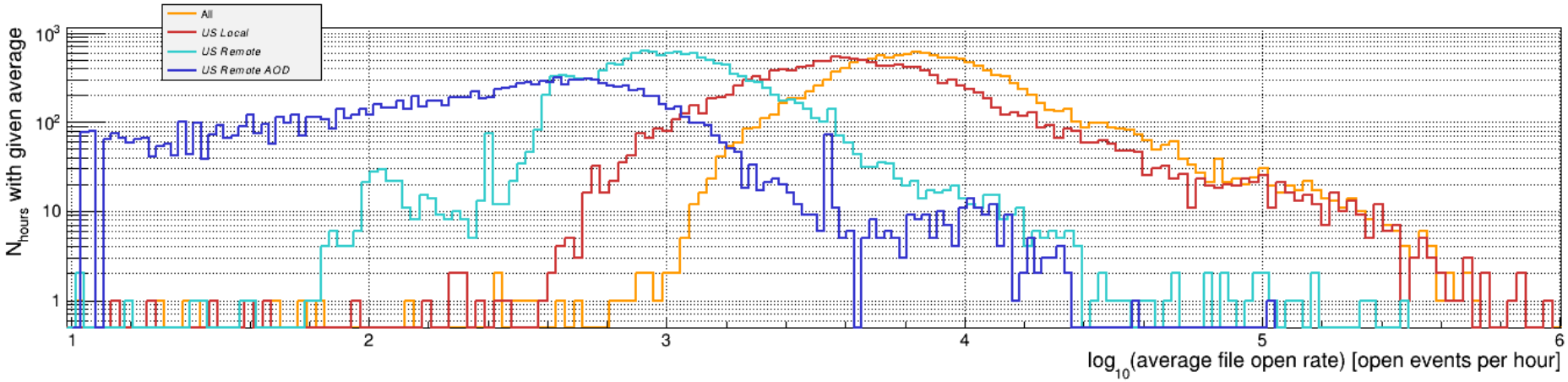
- about 750 users
- most transfers: pilot-cern/fnal (30/20%), 4 users at 5%
- servers: FNAL (60%), UCSD (15%), UFL (10%)
- clients: CNAF (30%), SINICS.TW (20%), CERN / DESY (15%), IC (10%)
- tiers: AODSIM (25%), RAW/GEN-SIM (20%), GEN (15%), AOD (7%)

5. Access from USA to elsewhere (0.3M):

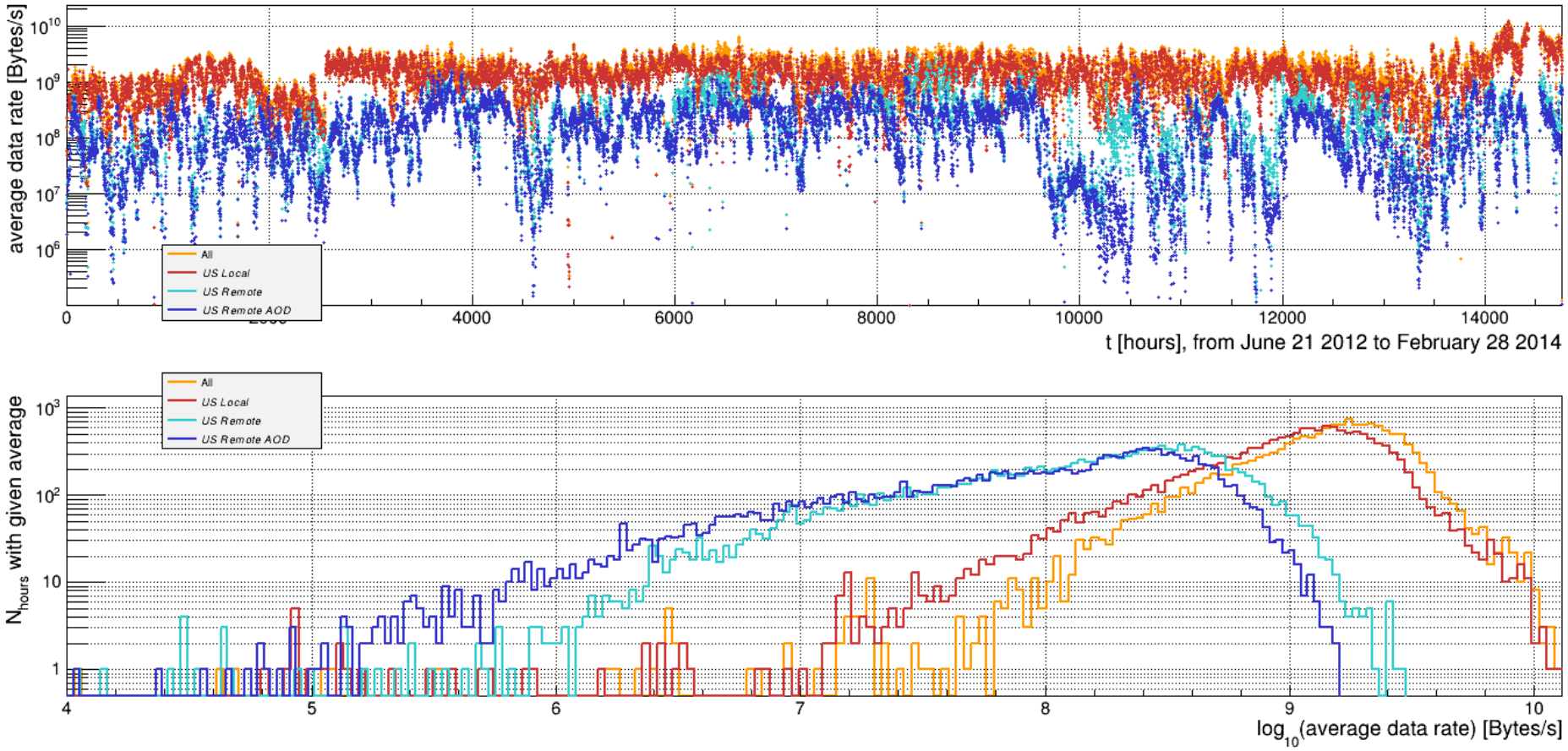
- 7% of traffic 4 min-bias GEN-SIM files, read from FNAL (3% of #)
- about 400 users
- cmspilot-cern, 50% of transfers, then 10 user between 2-10%
- servers: CNAF (50%), IN2P3 (10%), UCL/RAL/Pisa (above 5%)
- client: FNAL (75%), UW/Caltech (~10%)
- tiers: GEN-SIM (60%), AODSIM (25%), AOD (8%)

Total file-open rates

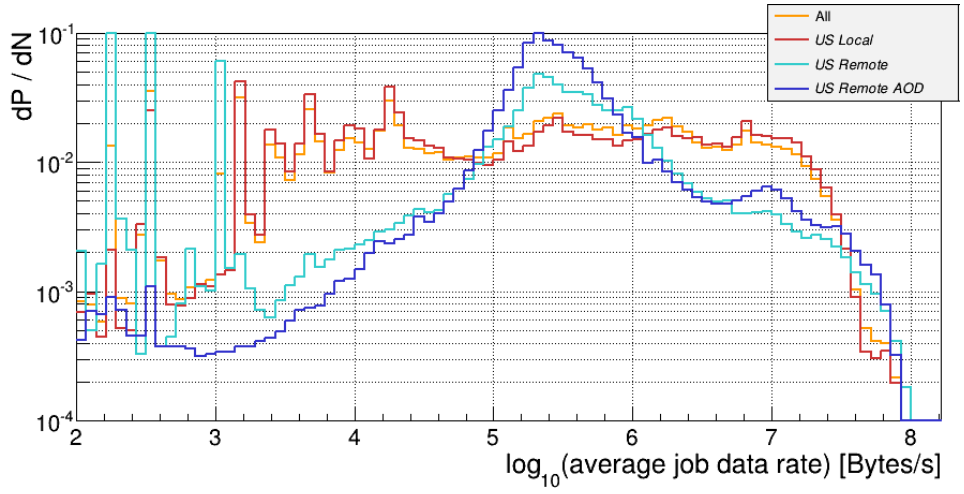


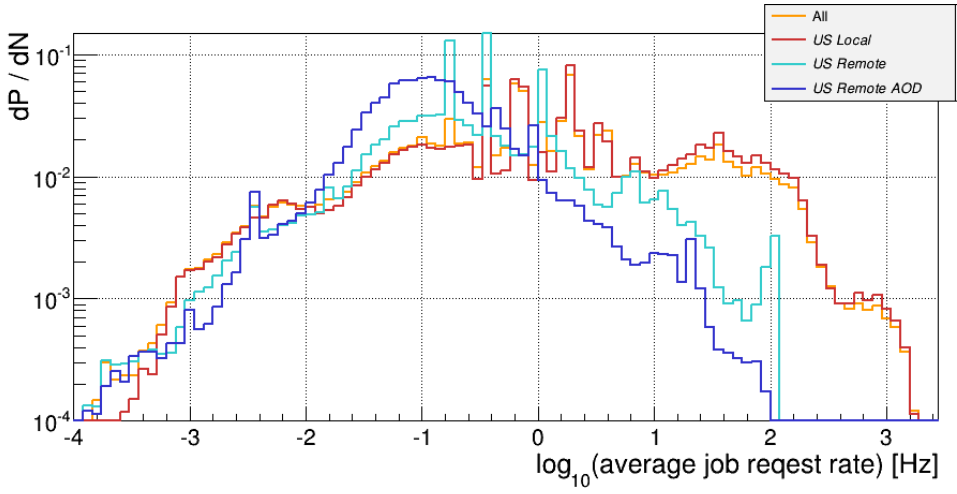


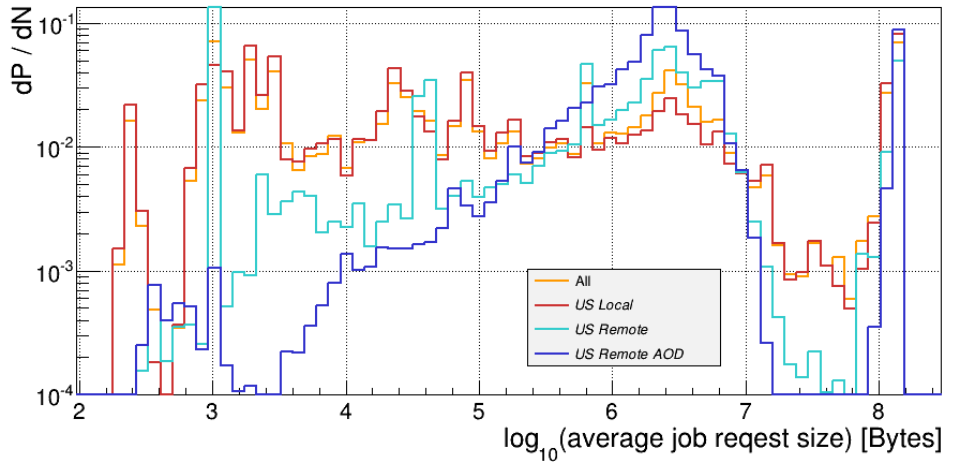
Total read rates

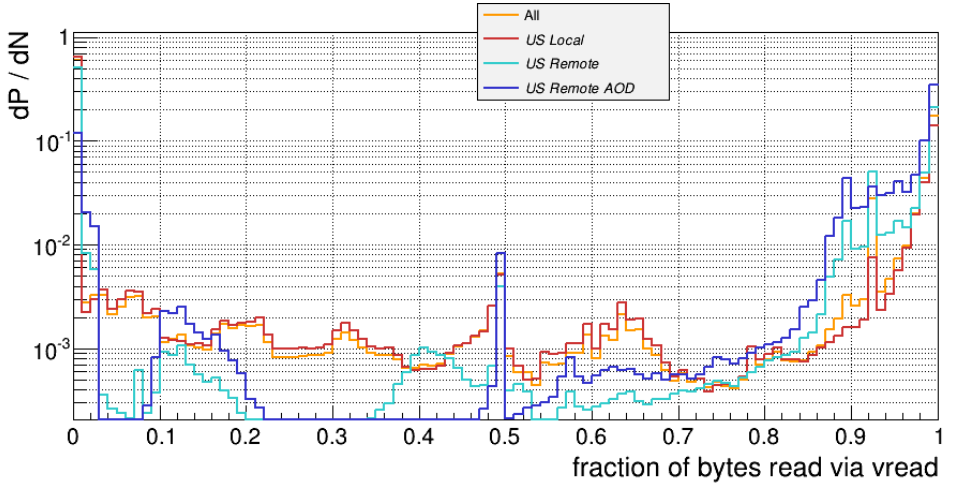


Comparison of job averages









AAA FAR data – what to really look at?

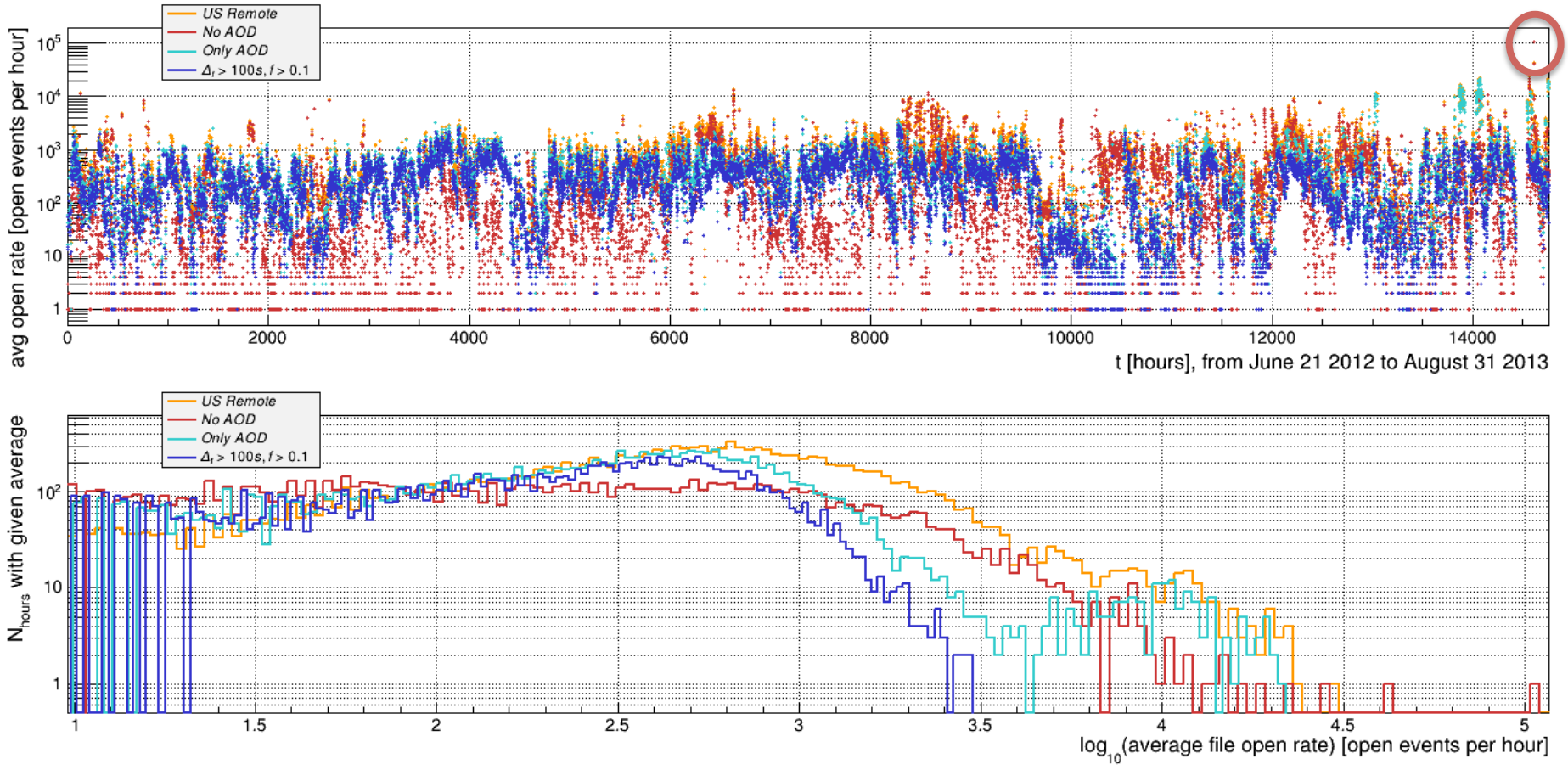
- Leave alone non-US data
  - too early, and I don't know what's going on there
  - there isn't that much of it anyway
- Also local access is problematic:
  - 50% unknown users, unknown job mixture
  - issues with job/site configuration
- This leaves as with remote access in the US:
  - Filter out known monitoring / testing users & paths

  This leaves about 8.2 M records!
  - Review general characteristics
  - Start focusing on AOD and AODSIM access – **~4.3 M**

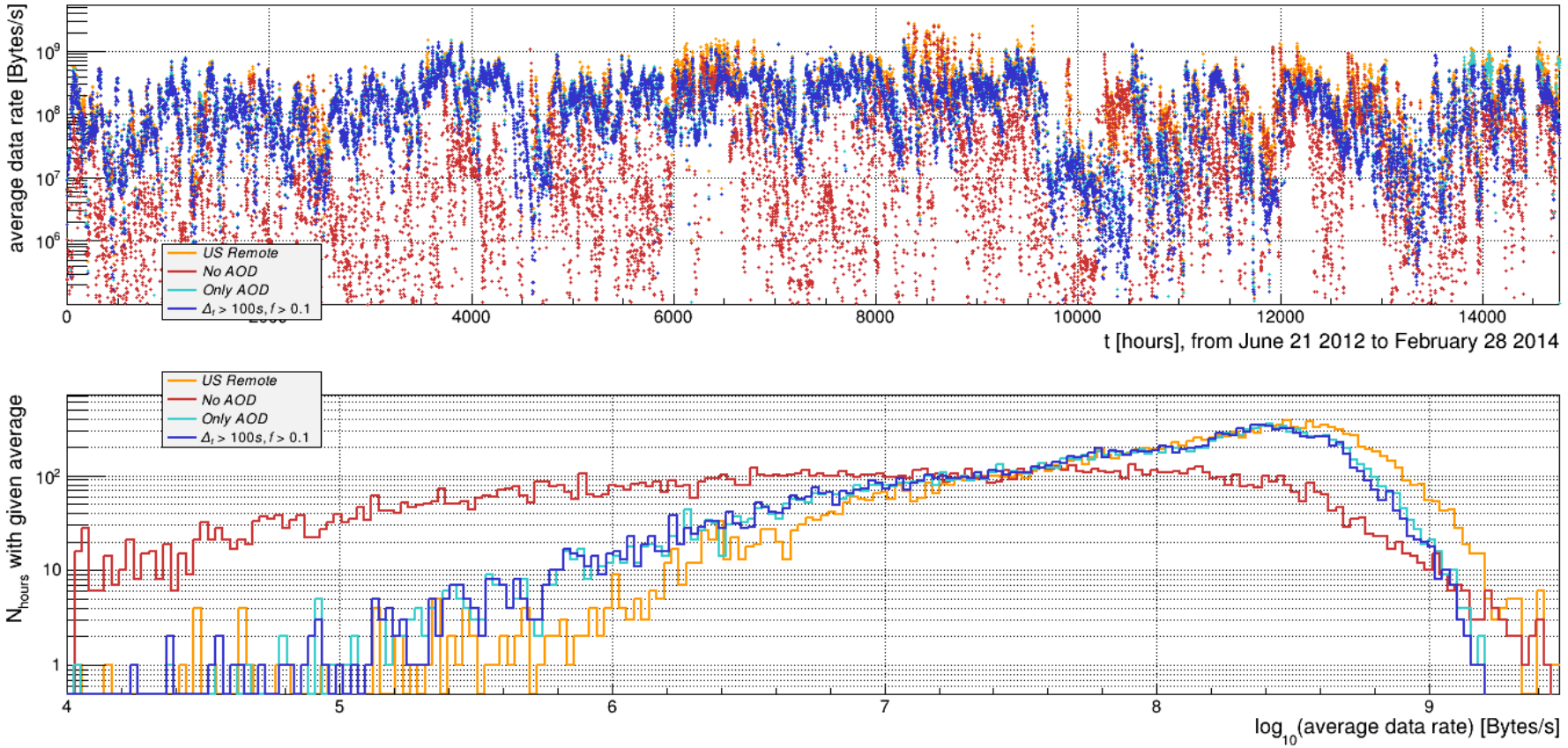
1. Filter out non interesting things from before
2. Compare AOD vs. non-AOD, look at access from user directories
3. Accumulate total file-open & transfer rates for the federation
4. Plot histograms: data rates, read/vec read fractions, duration, …
5. Plot 2D histograms, everything against everything and vs. time!

STEP II. – FOCUS ON US REMOTE DATA ACCESS

Total US remote file-open rates

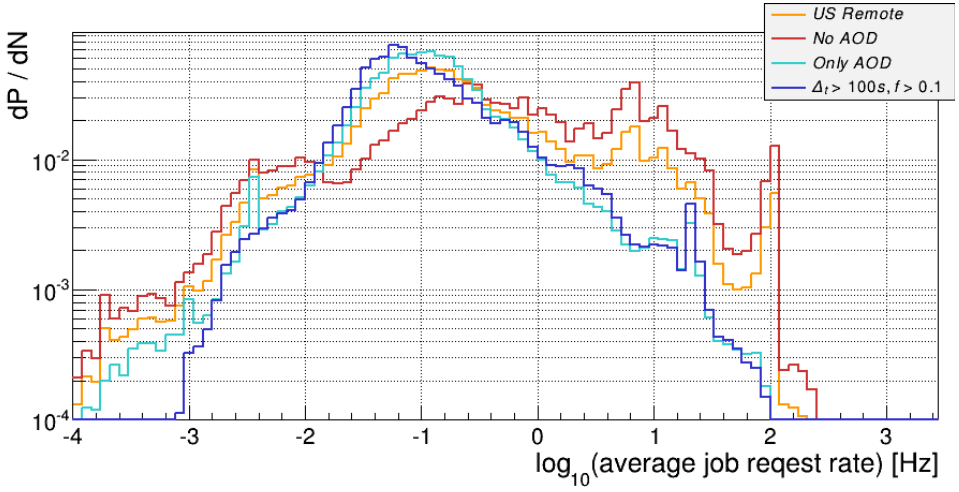


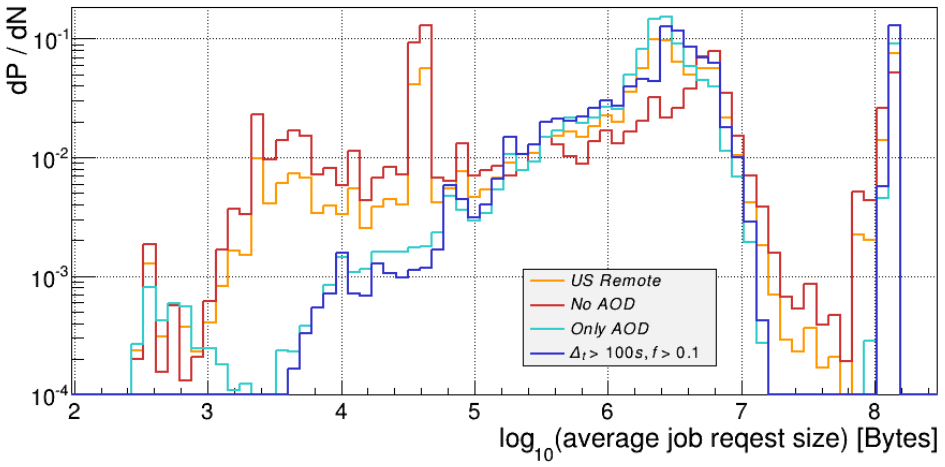
Total US remote read rates



Job request rates and sizes

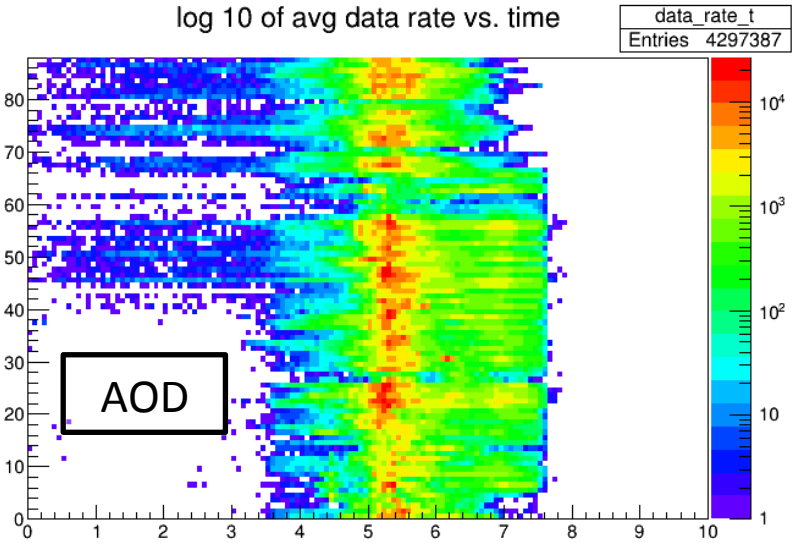
- This goes over orders of magnitude, log-log!
- AOD access better behaved, users do wonders
- Notice the peak at 128 MB request size – these are xrdcp / lazy preloads

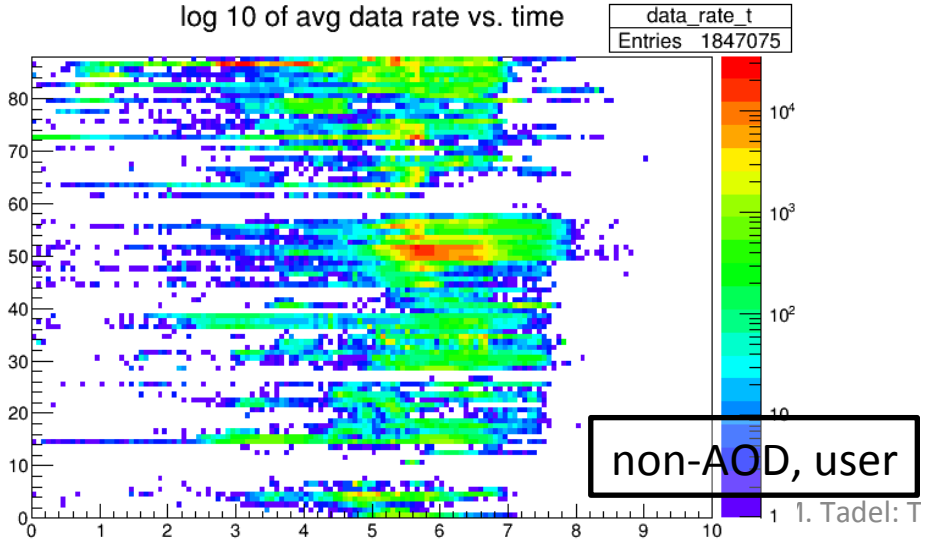


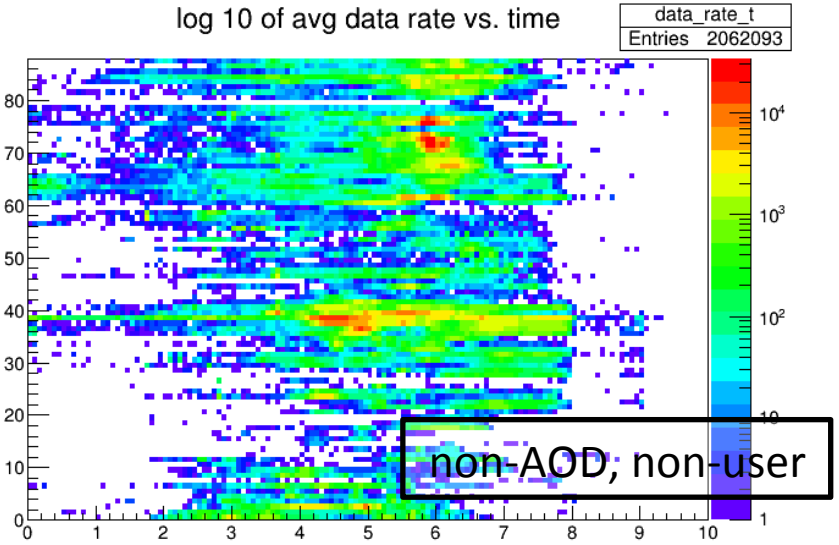


Job read rate vs. time in weeks

- AOD the most consistent
- User areas the most chaotic, in read rate and in time



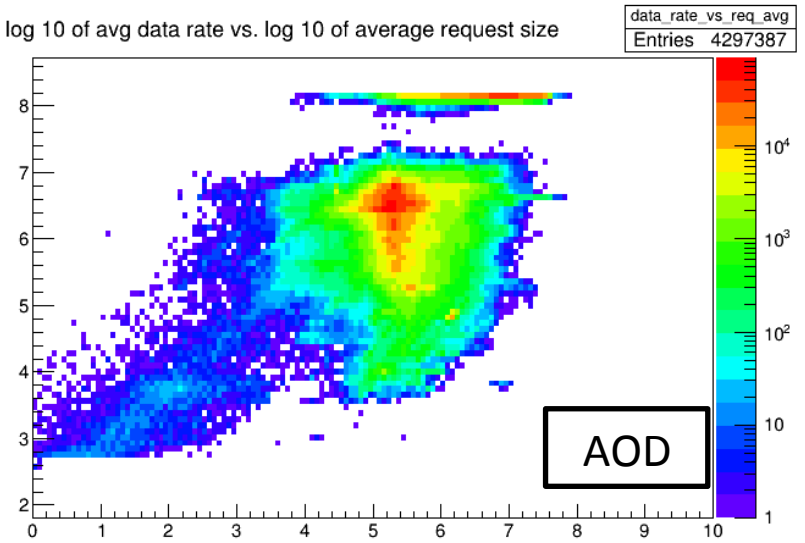


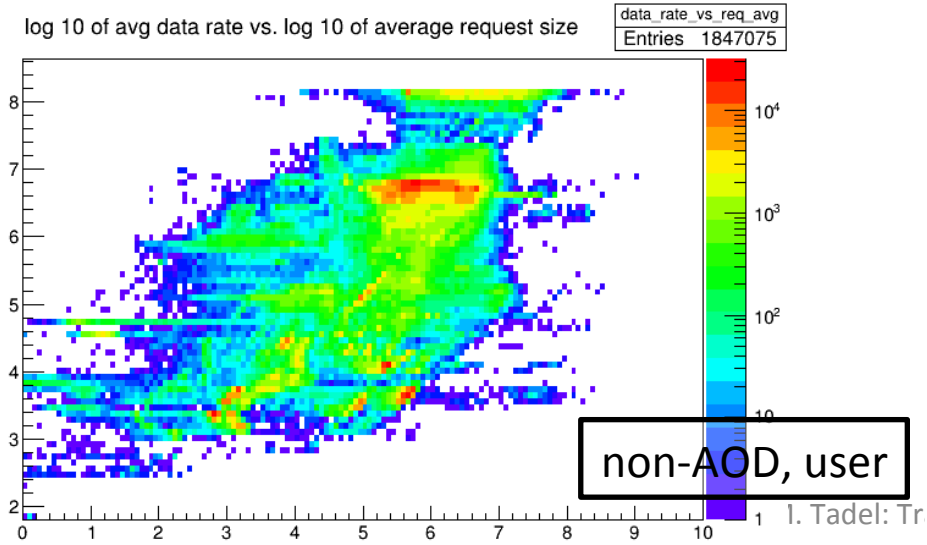


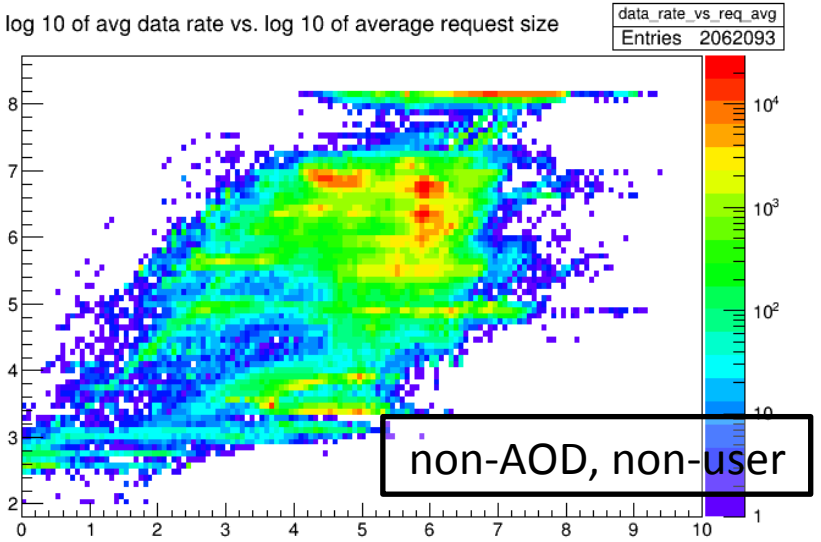
1. Tadel: T

Job read rate vs. request size

- Notice xrdcp / lazy-preload peaks
- Again, AOD access consistent, +/- an order of magnitude ☺

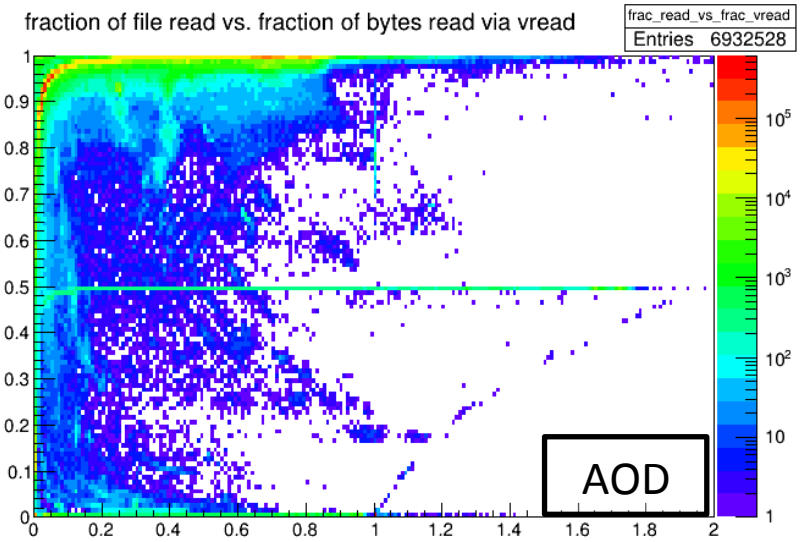


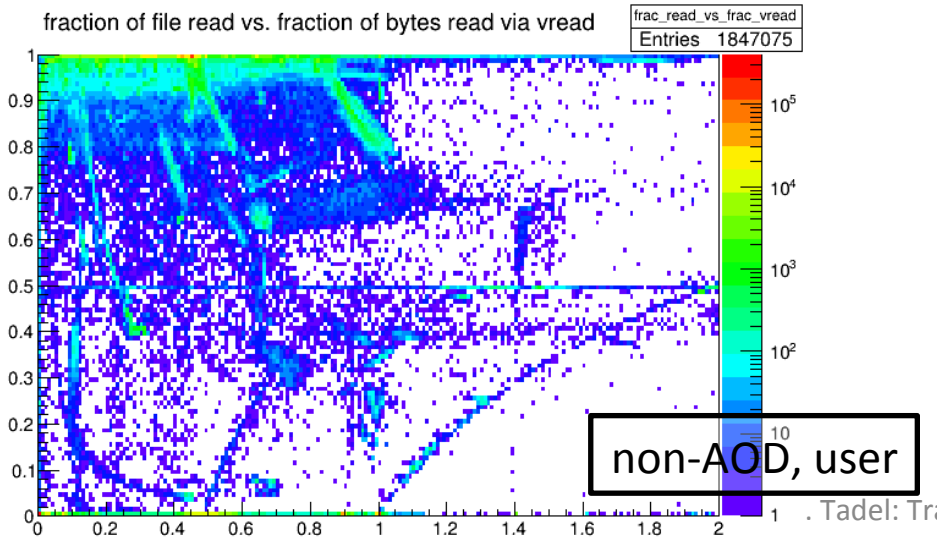


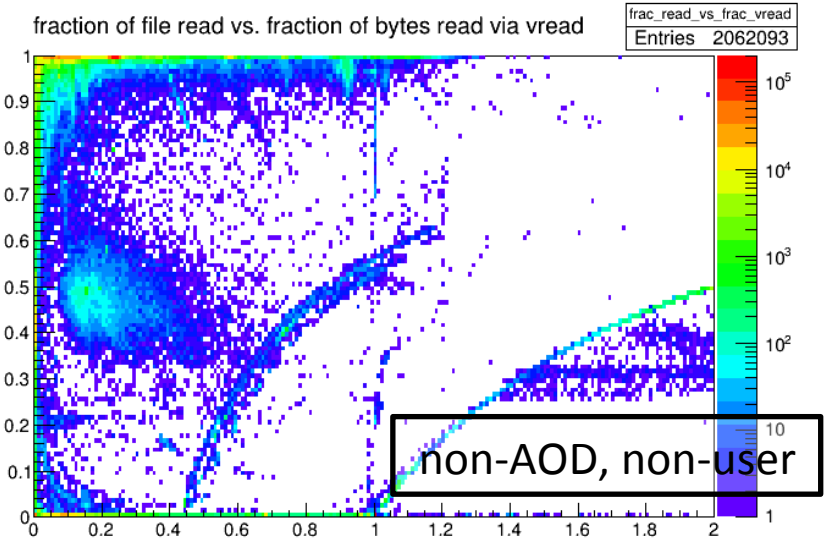


Job fraction read vs. fraction of vector reads

- This is the plot where one would catch bad guys
- CMS is doing good ☺
- The ridge at 50% vector-read is a monitoring bug in the first 20% of data







1. The dataset
2. Request size & offset distributions
3. Vector reads

STEP IV. – ANALYSIS OF IOV DATA: SEEKS AND OFFSETS

AAA **IOV** data overview I.

2013 Feb - 2014 Feb

3.4×10^7s = 9,421 h = 56.1 weeks = 12.9 months

	N (k)
All	2,364
Without monitoring/tests	785
US	601
local AOD	103
remote AOD	303
idem without xrdcp	260
idem with bad entries out	193

matevz@desire xrd-far> ls --format single-column –sh

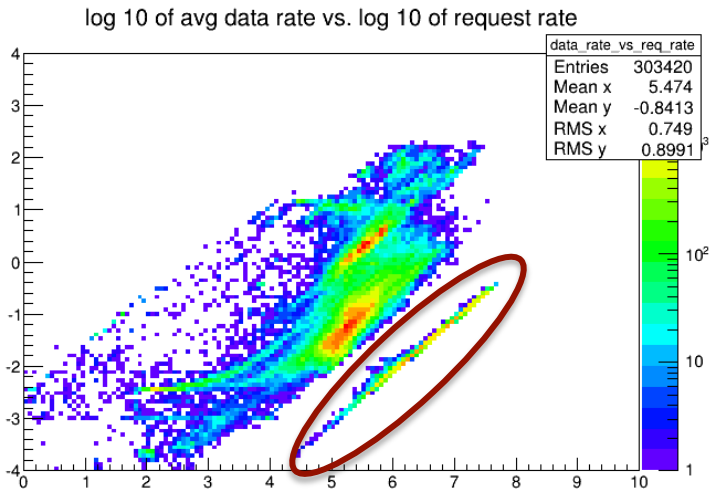
total 41G	0.8G xmxxx-2013-08.root
3.9G xmxxx-2013-02.root	1.5G xmxxx-2013-09.root
5.4G xmxxx-2013-03.root	3.4G xmxxx-2013-10.root
4.6G xmxxx-2013-04.root	6.7G xmxxx-2013-11.root
2.0G xmxxx-2013-05.root	2.2G xmxxx-2013-12.root
4.6G xmxxx-2013-06.root	3.5G xmxxx-2014-01.root
1.0G xmxxx-2013-07.root	1.7G xmxxx-2014-02.root

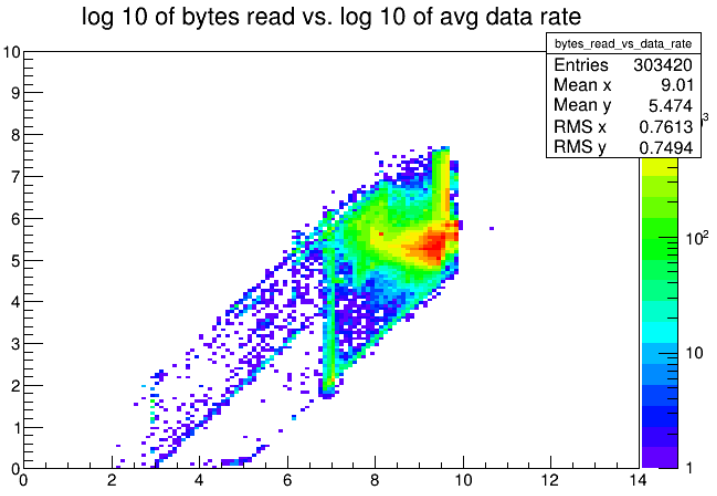
18,500 Bytes / record
(140-times FAR size)

Volume: 81% AODSIM, 9% AOD
→ together 800 TB

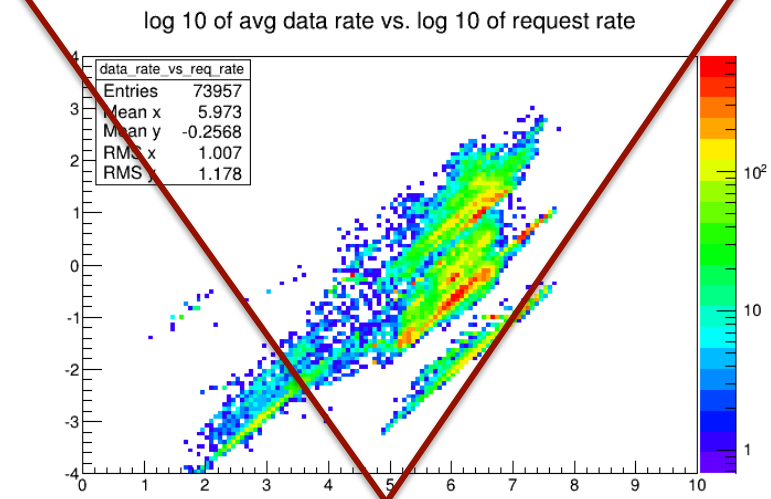
AAA **IOV** data overview II.

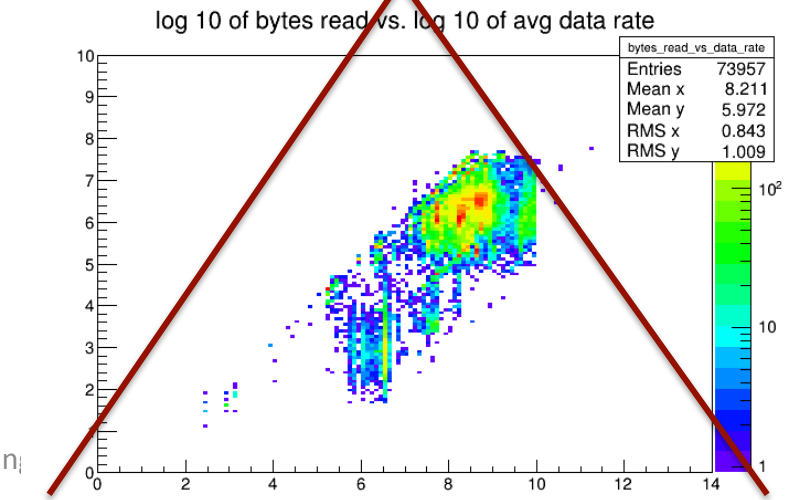
AOD remote





User directories





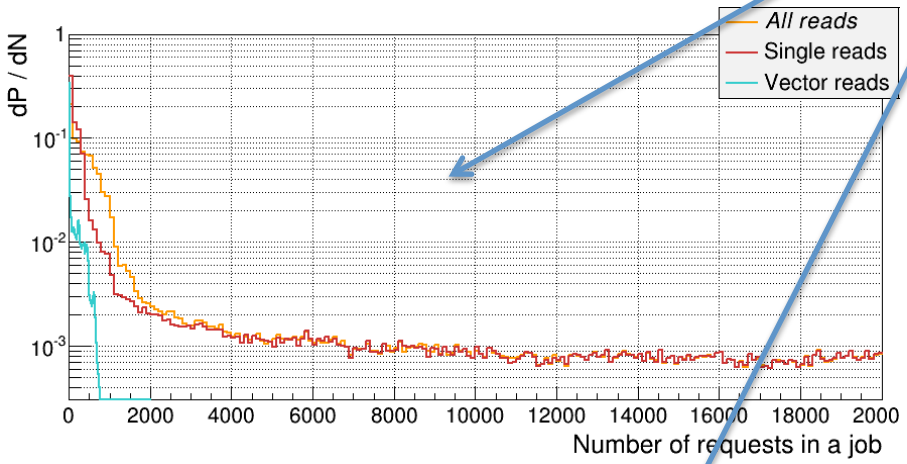
Things get complicated here ☺

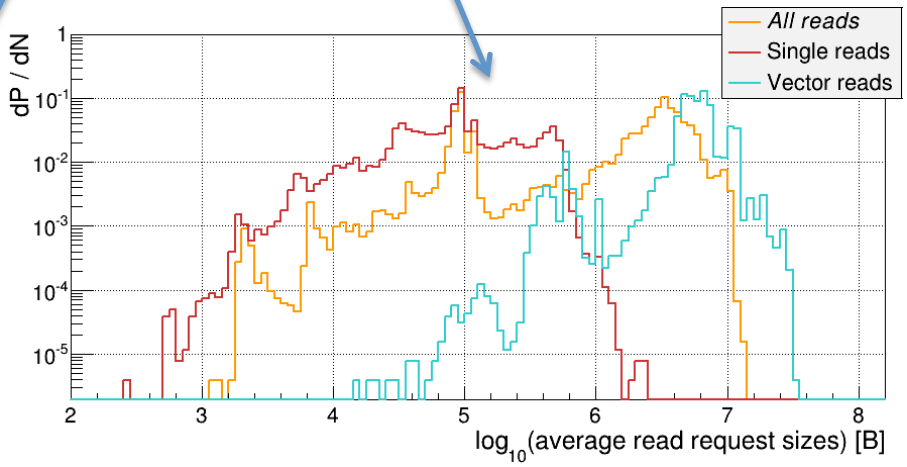
Read request size & offset distribution

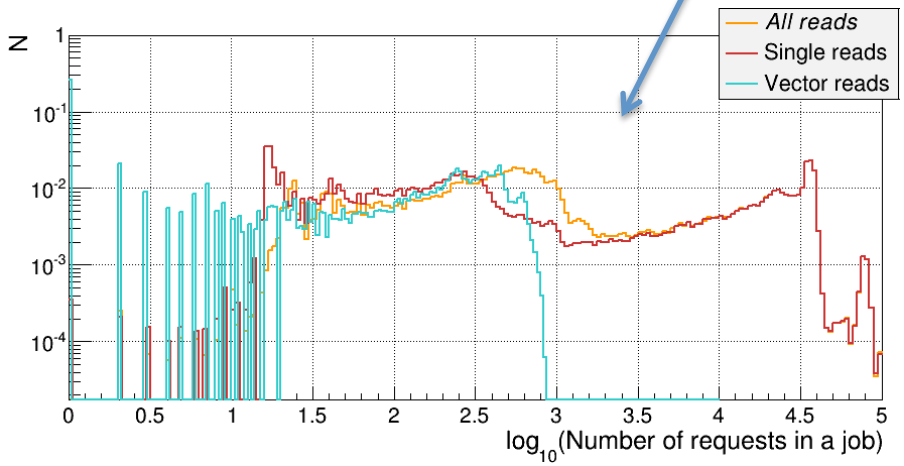
- First, let's look at full read requests
  - vector read is counted as one entity, we'll look at offsets within vector reads soon
- Histograms that will be shown:
  - per-job quantities: average, sum, (number, min, max)
    - expressed in bytes and in file-size fraction
  - cumulative distributions – each seek / read is an entry
  - separate positive / negative seeks
  - separate seeks from single/vector reads, preceded by single/vector reads

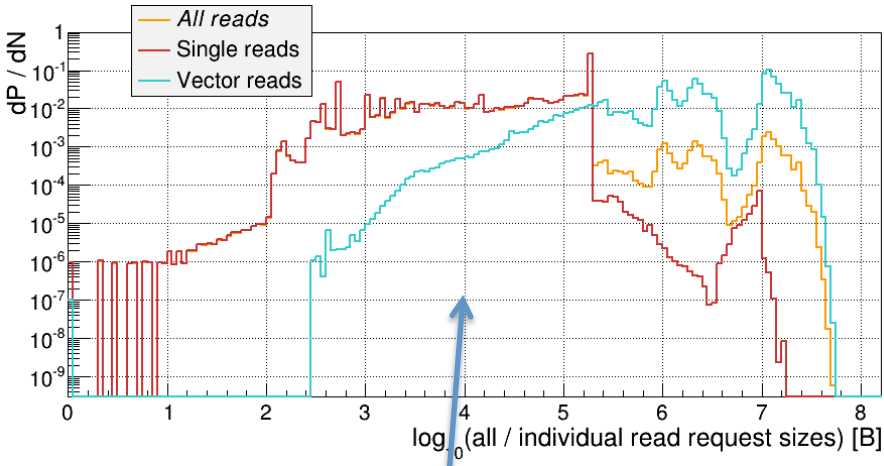
E.g., read requests:

One entry per job – 190 x 10^3









One entry per read request – 1.7 x 10^9

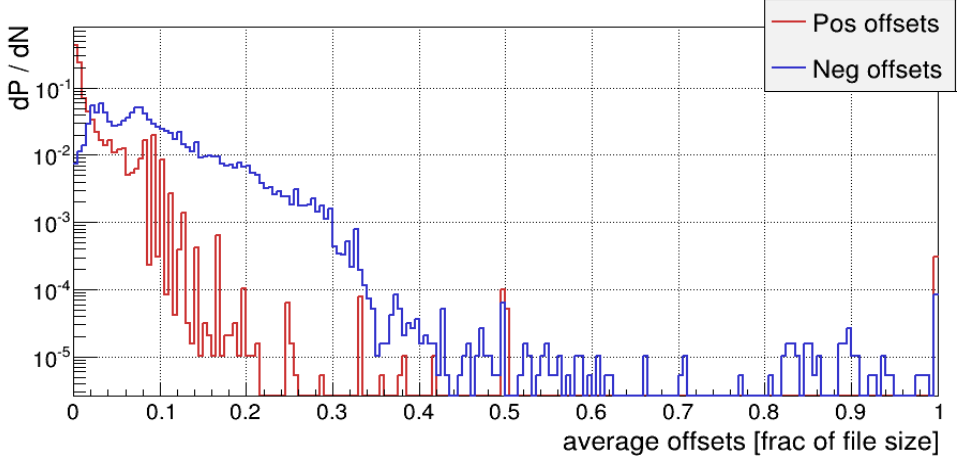
Positive and negative offsets

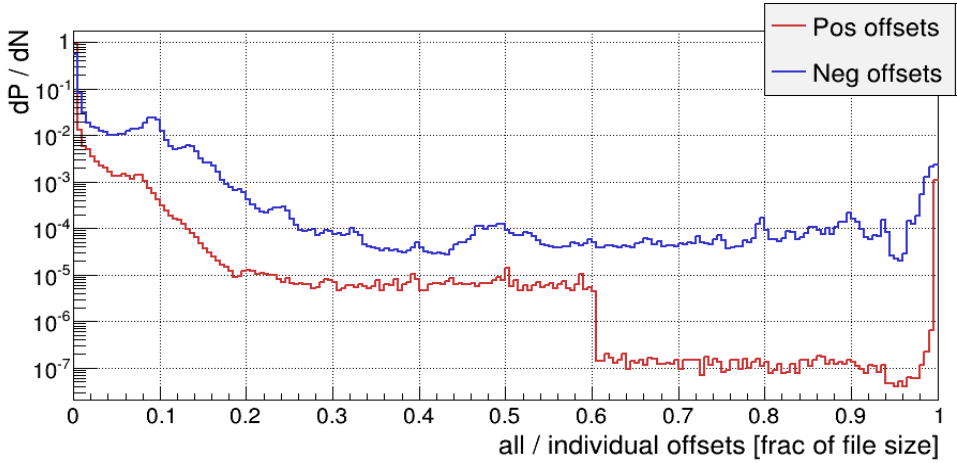
This is between reads:

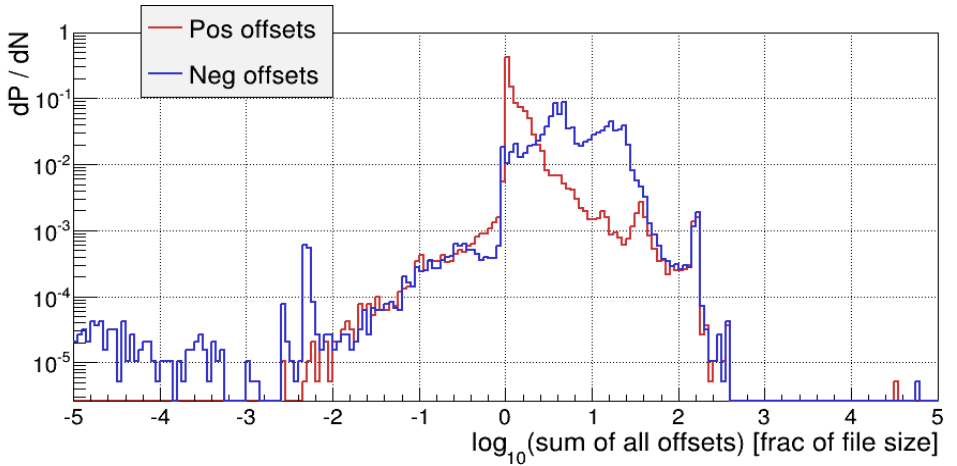
- a single read is an implicit seek forward, not counted
- a vector read is an implicit seek forward to the end of the last sub-request – we'll look at intra-offsets soon

Note the log x-axis on the last plot:

- sum of seeks per job → we typically seek backwards many times the file size!





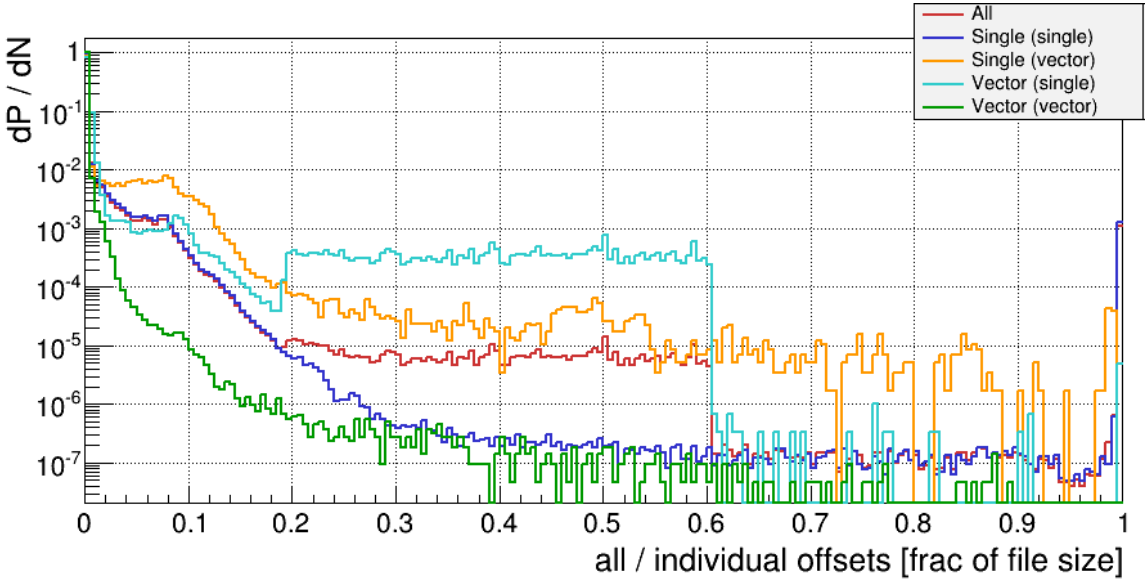


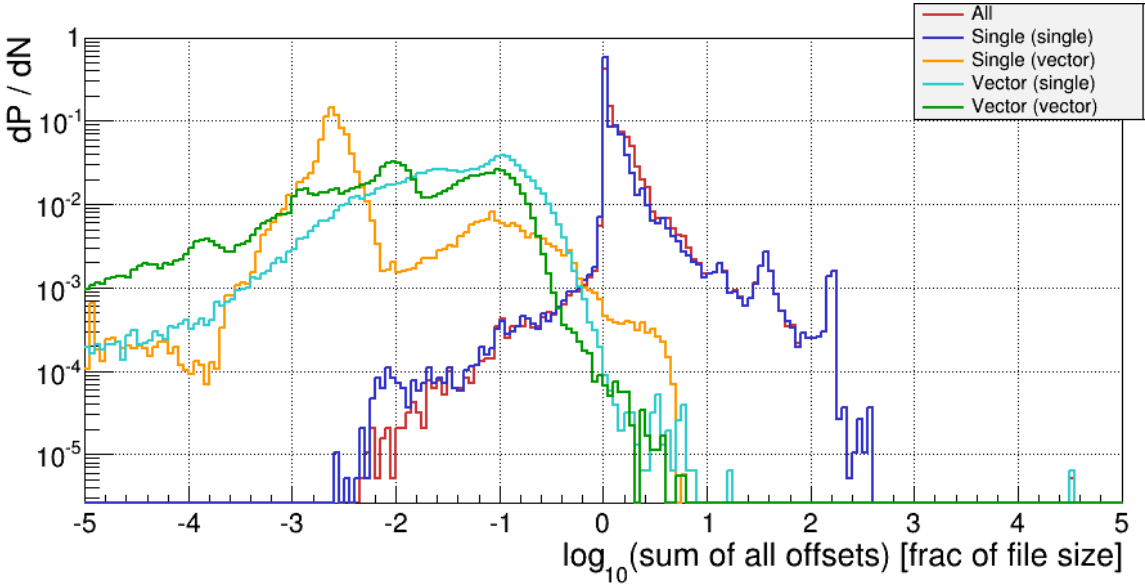
Positive offsets

Compare offsets for single versus vector reads.

Separate them also based on what the previous read was (single / vector).

Single reads jump forward more – especially when preceded by a single read.





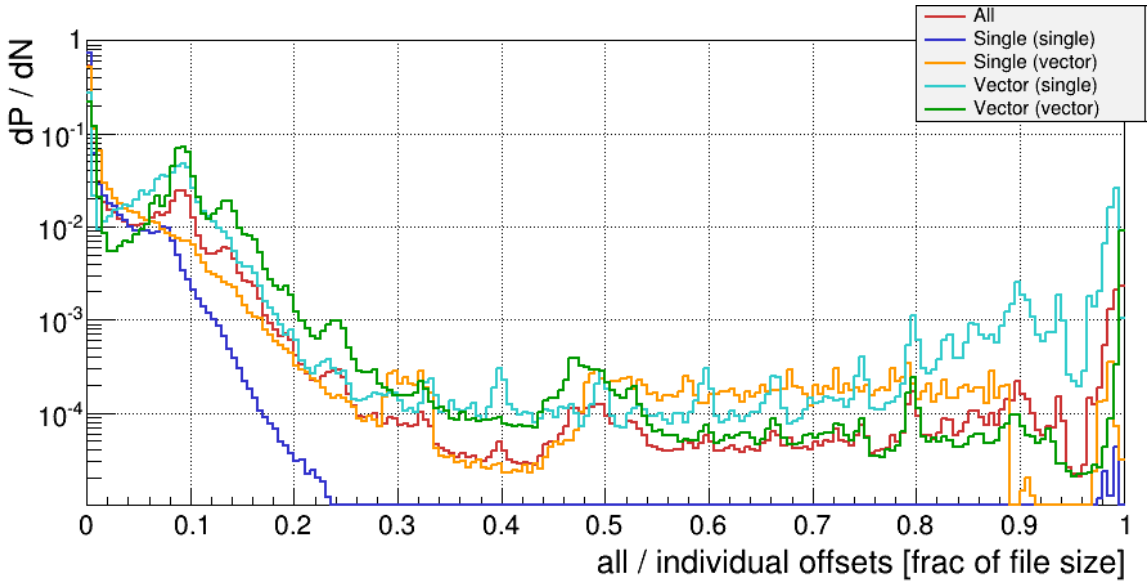
Negative offsets

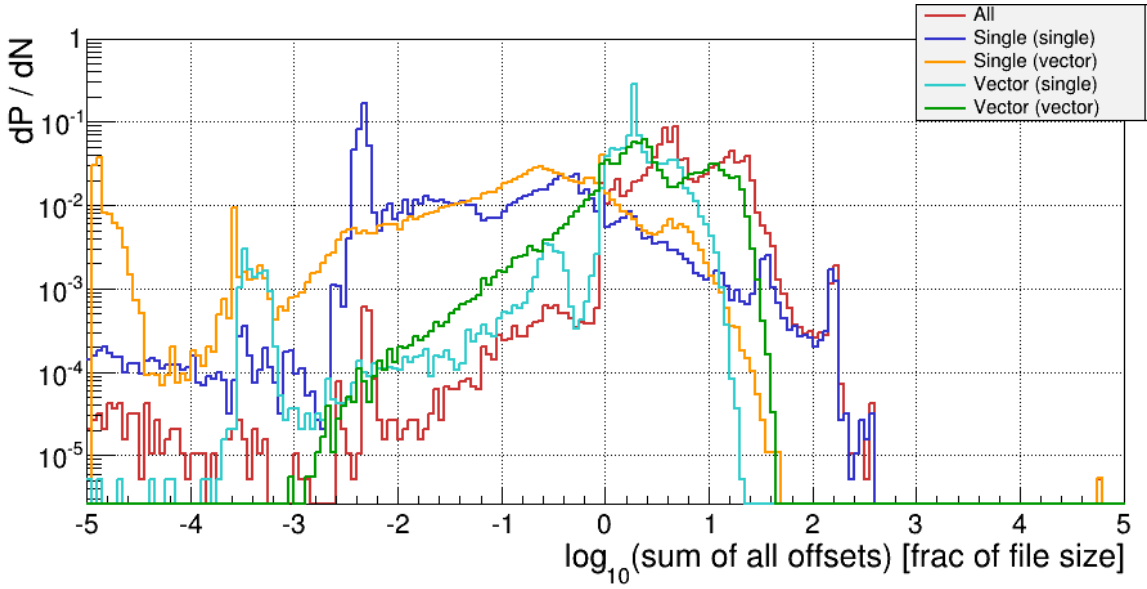
Compare offsets for single versus vector reads.

Separate them also based on what the previous read was (single / vector).

Vector reads jump backwards more likely,

but in sum, there is also significant contribution from single reads, esp. for cases with “large backwards motion”

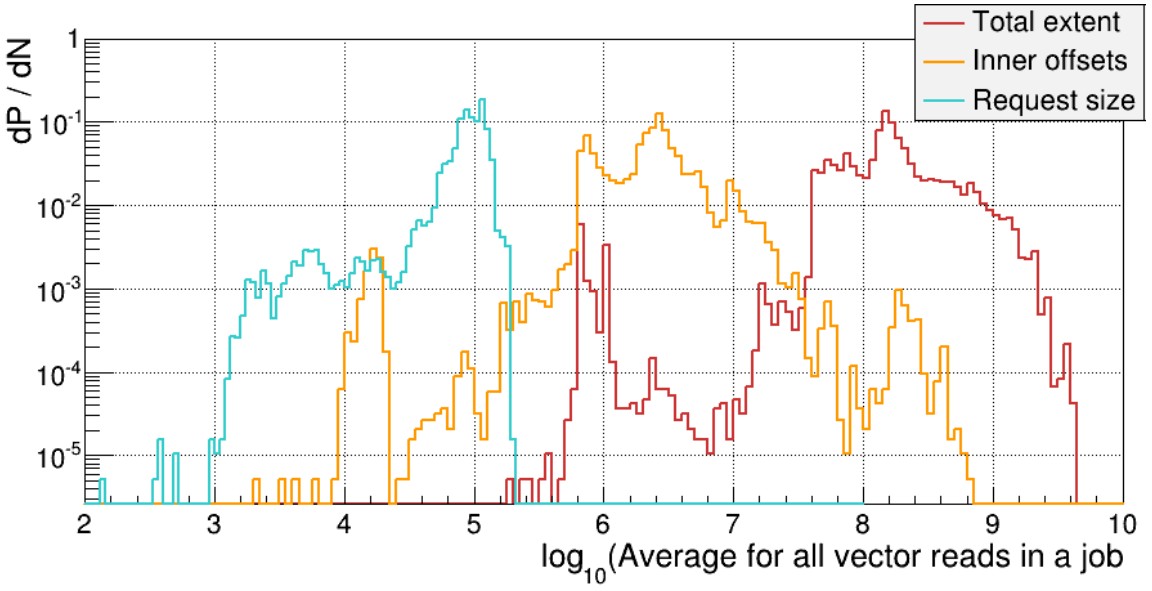




Offsets & extents *within* vector reads

Averages (in bytes):

- requests: 10 kB
- offsets: 1 – 10 MB
- total extent: up to 1 GB

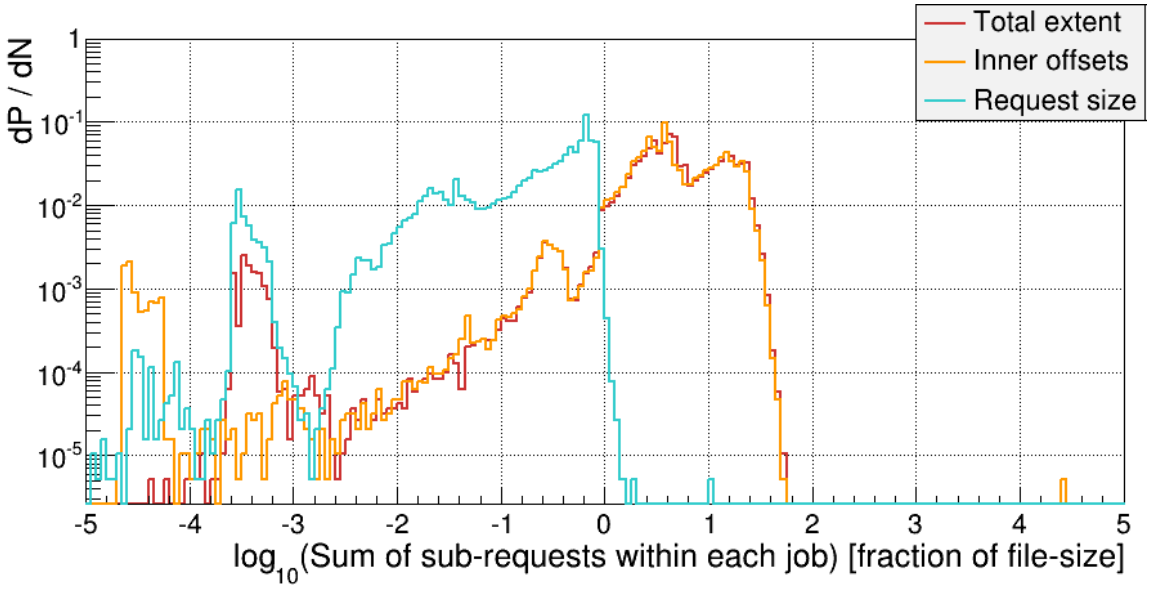


Sum of each:

- requests: sum up to at most the file size
- offsets and total extent practically the same:

  They add up to from a couple to 20-times!

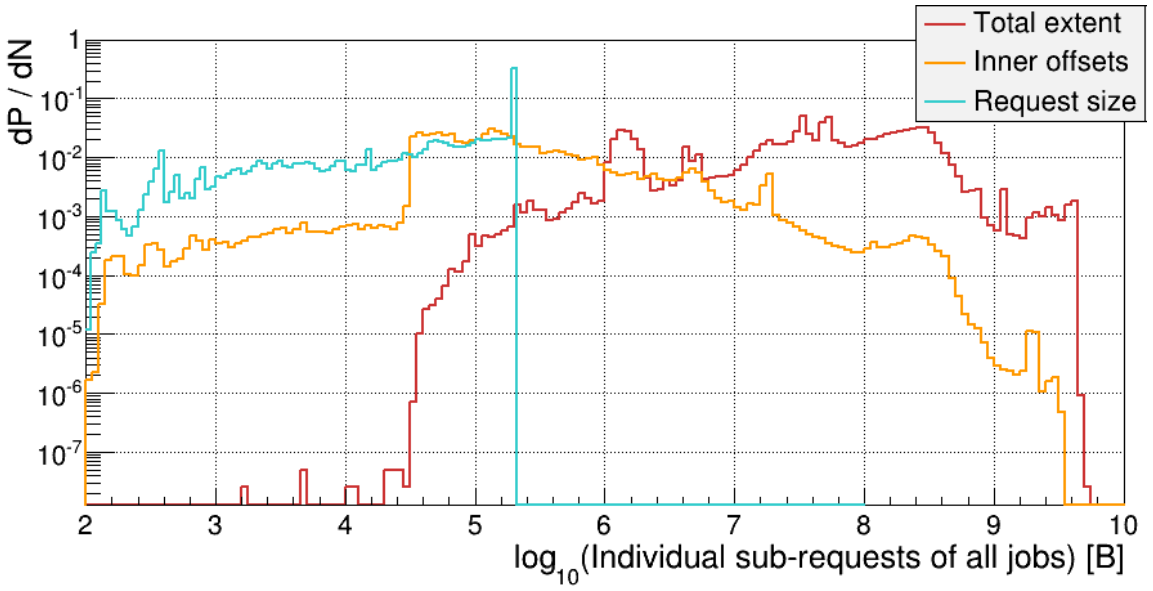
This covers the "missing" positive offsets.



And individual vector sub-requests

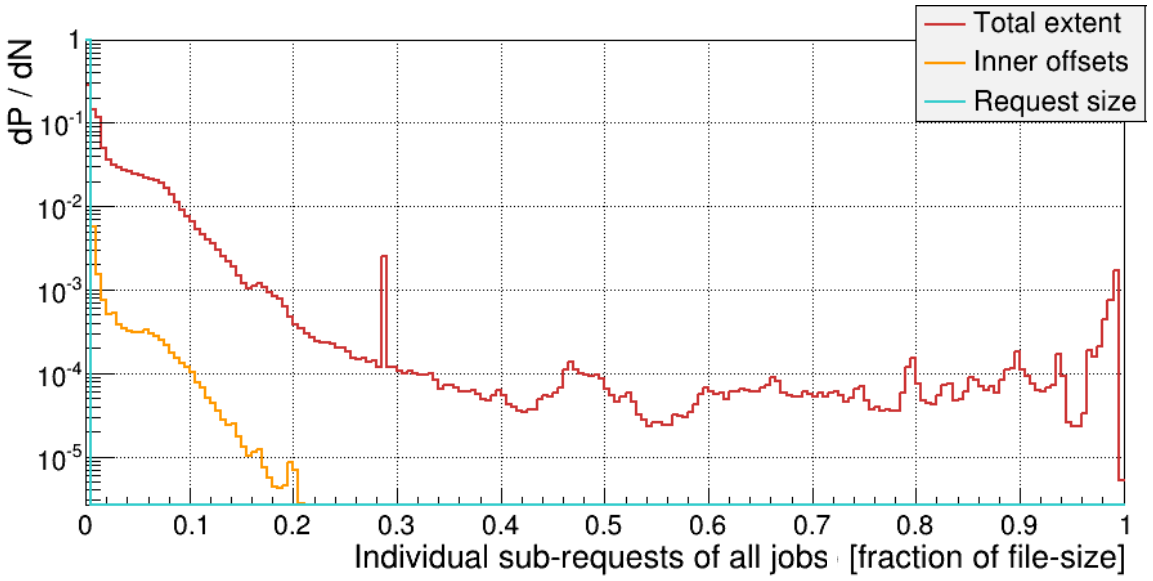
In bytes:

- Sharp cutoff at 20 kB
  CMS buffer sizes



In fraction of file size:

- Offsets don't cross 20% of file size.



1. Is all this really relevant?
2. Is there a killer metrics for things that matter?
3. The last slide

TIME TO SLOW DOWN …

Winding down …

- There is a lot one can extract from XRootd monitoring alone
  - saying this with my XRootd hat on 
  - the plots are
    - of particular interest to CMS
    - and of general interest to other VOs
  - BUT … things are about to change, for CMS, too
- All this will be of limited value soon
  - What matters is that we can redo this easily

What's the message?

- This is a large selection of plots to look at
  - and one has to weed out a lot of noise
- One could redo the analysis regularly … or make dedicated tests with relevant workloads.
- But … we're really after efficient remote access:
  - CPU efficiency (tied with experiment IO stack)
  - Computing model
    - Are we doing remote access for a single job?

A killer metrics?

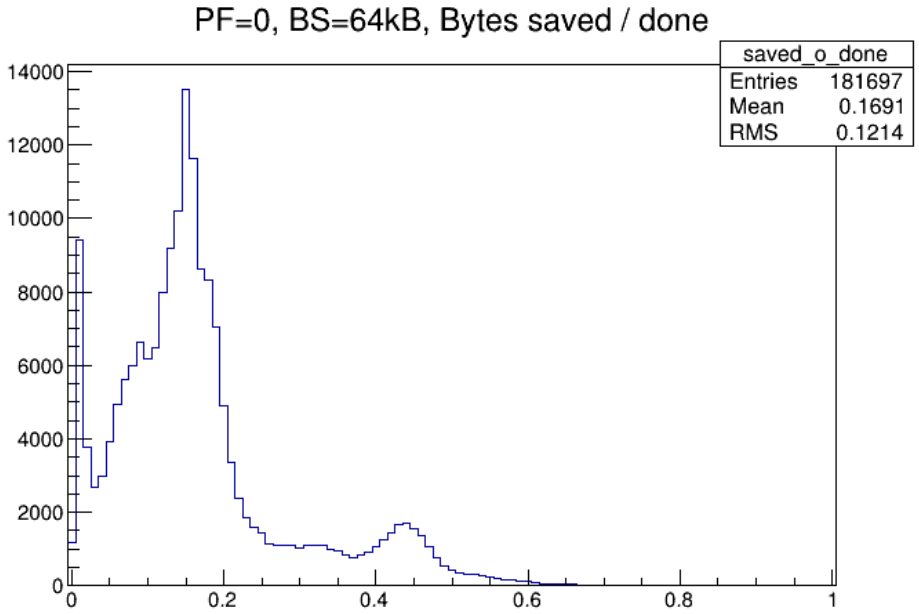
- Part of motivation was understanding how to best implement & configure a caching proxy:
  - proxy can do prefetching, and
  - can store data for later use.
- If computing model envisions on-demand data placement, the remote-access-while-reading only happens once

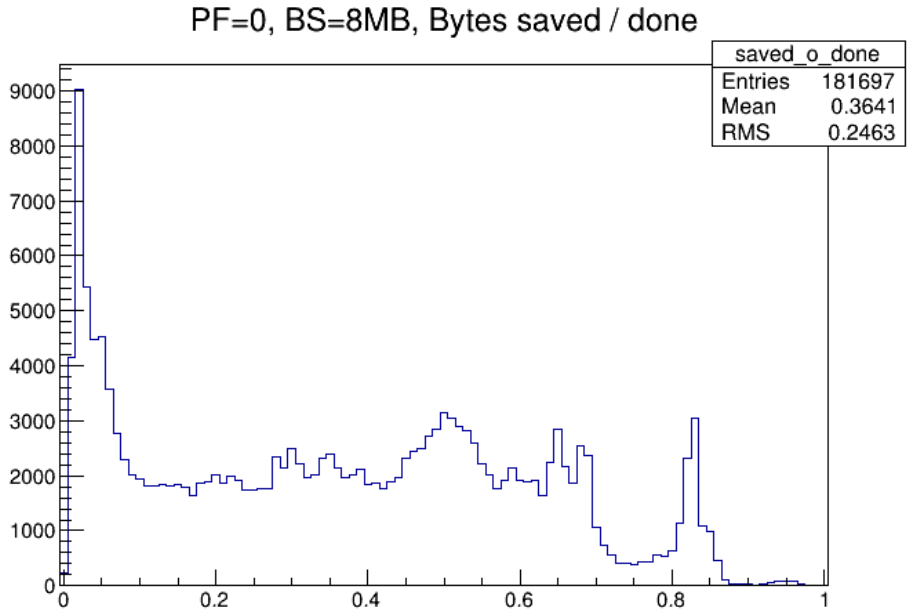
A caching proxy simulation!

Caching proxy simulation preview

- More about this in XRootd caching proxy talk
- Can choose block size and prefetching rate
  - play the IOV data through
  - then observe the performance
- 100 Gbps networks are coming this summer!

Fraction of data served from cache





The last slide …

- There are a lot of things that we do not want see in access reports.
- Analysis is complicated because of the above and requires “closeness” to the federation.
- Things change every week, +/-.

But … there are tools to study various remote access strategies

- So we can put them to use when big changes hit.
- Interesting (crucial?) for non-HEP VOs considering remote data access.